Electronic Transactions on Numerical Analysis.
Volume 46, pp. 447–459, 2017.

ISSN 1068–9613.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

COMPUTING THE EIGENVALUES OF SYMMETRIC TRIDIAGONAL MATRICES VIA A CAYLEY TRANSFORMATION*

JARED L. AURENTZ*, THOMAS MACH†, RAF VANDEBRIL‡, AND DAVID S. WATKINS§

Dedicated to the memory of Bill Gragg.

Abstract. In this paper we present a new algorithm for solving the symmetric tridiagonal eigenvalue problem that works by first using a Cayley transformation to convert the symmetric matrix into a unitary one and then uses Gragg's implicitly shifted unitary QR algorithm to solve the resulting unitary eigenvalue problem. We prove that under reasonable assumptions on the symmetric matrix this algorithm is backward stable. It is comparable to other algorithms in terms of accuracy. Although it is not the fastest algorithm, it is not conspicuously slow either. It is approximately as fast as the symmetric tridiagonal QR algorithm.

Key words. eigenvalue, unitary QR, symmetric matrix, core transformations, rotations

AMS subject classifications. 65F15, 65H17, 15A18, 15B10,

1. Introduction.

In the past few decades a considerable number of fast and stable algorithms have been developed for the solution of the symmetric matrix eigenvalue problem. Arguably no other class of matrices has enjoyed such success, with one exception, unitary matrices. Unitary matrices share many of the same properties as symmetric ones: normality, localized eigenvalues, and special condensed forms, which allow to represent symmetric and unitary matrices with two vectors. These similarities often allow one to immediately adapt the symmetric algorithms to the unitary ones.

In this paper we will work in the other direction, albeit in a roundabout way. Instead of developing a new algorithm that works directly with the symmetric matrix, we first convert the symmetric matrix into a unitary one via a Cayley transformation and then solve the unitary eigenvalue problem using Gragg's unitary QR algorithm [8].

Moving between symmetric and unitary matrices using Cayley transformations is not a new idea. Perhaps the first instance involving the spectrum of an operator appeared in 1930 when von Neumann used a Cayley transformation to prove the spectral theorem for unbounded self-adjoint operators [11]. In 1968 Franklin used Cayley transformations to count the eigenvalues of a general matrix that lie in a half-plane [6]. More recently, Gemignani applied a Cayley transformation to turn a unitary eigenvalue problem into a symmetric one, which is ultimately where our inspiration comes from [7].

*Received February 29, 2016. Accepted October 30, 2017. Published online on November 10, 2017. Recommended by Martin Gutknecht. This research was supported through the program "Research in Pairs" by the Mathematisches Forschungsinstitut Oberwolfach in 2016. The research was also partially supported by the Research Council KU Leuven, projects CREA/13/012 Can Unconventional Eigenvalue Algorithms Supersede the State-Of-The-Art, C14/16/056 Invers-free Rational Krylov Methods: Theory and Applications, PFV/10/002 Optimization in Engineering (OPTEC), by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), and by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement no. 291068. The views expressed in this article are not those of the ERC or the European Commission, and the European Union is not liable for any use that may be made of the information contained here.

*Instituto de Ciencias Matemáticas, Campus de Cantoblanco UAM, Calle Nicólas Cabrera, 13-15, 28049 Madrid, Spain (jared.aurentz@icmat.es).

†Department of Mathematics, School of Science and Technology, 53 Kabanbay Batyr Ave, 010000 Astana, Kazakhstan (thomas.mach@nu.edu.kz).

‡KU Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Leuven (Heverlee), Belgium (raf.vandebril@cs.kuleuven.be).

§Department of Mathematics, Washington State University, Neill Hall, Pullman, WA 99164-3113, USA (watkins@math.wsu.edu).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

The curious reader might be thinking *Why would one do this?* Ultimately we did it because we could. First, a fast and stable version of Gragg's algorithm is now freely available as part of the Fortran 90 library eiscor [2, 3]. Previously such an implementation was not available, and now that it is, it seems only fitting to put it to good use. Second, we will show in this paper that transforming between symmetric and unitary problems can be done stably and efficiently, which leads to a new stable algorithm for solving symmetric tridiagonal eigenvalue problems. It is about as accurate as other well-known algorithms, and it is faster than one might expect. Although it is not the fastest algorithm available, it is comparable in speed with the symmetric, tridiagonal QR algorithm.

The paper is organized as follows. In Section 2 we discuss the main idea of using a Cayley transformation to transform a symmetric eigenvalue problem into a unitary one and then state our algorithm. We also discuss the details of our implementation and show that the complexity is $O(n^2)$ where n is the size of the matrix. In Section 3 we give a proof of backward stability. In Section 4 we perform some numerical experiments that illustrate the speed and accuracy of our algorithm. Finally, in Section 5 we give our concluding remarks.

Unless stated otherwise the following are assumed true. The matrix T is real, symmetric, and tridiagonal. The matrix I is the identity. The letter u denotes the unit round off. The letter i is the imaginary unit, $i^2 = -1$. If A is an $n \times n$ matrix we denote the entry of A in row j and column k by A_{jk}. We denote by $\overline{A}$ the matrix whose entries satisfy $\overline{A}_{jk} = \overline{A_{jk}}$, and we denote by A^T the matrix whose entries satisfy $(A^T)_{jk} = A_{kj}$ for all $j, k = 1, \ldots, n$. Similarly we define A^H by the identity $A^H = \overline{A}^T$.

2. Solving the symmetric eigenvalue problem. Let φ be the Cayley transformation[1]

$$\varphi(z) = (i - z)(i + z)^{-1}.$$

The function φ maps the interval $[-1, 1]$ to the semicircle $\{z = e^{i\theta} : \theta \in [-\pi/2, \pi/2]\}$ and all other points on the real line into the remainder of the unit circle. Figure 2.1 illustrates φ graphically using a few isolated points.

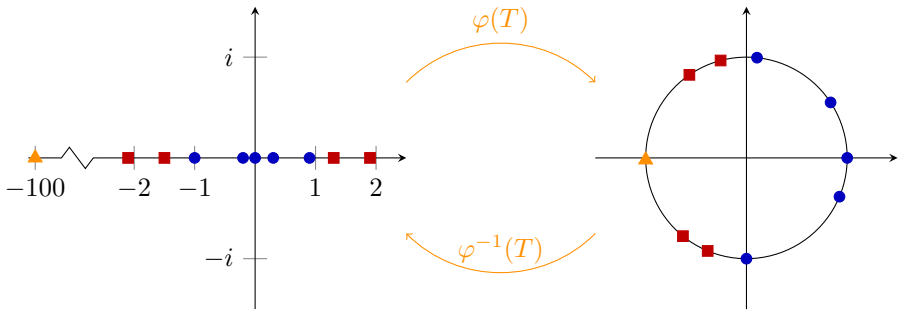


FIG. 2.1. *Illustration of how φ maps points from the real line to the unit circle. Points in $[-1, 1]$ (dots) are mapped to the right half of the unit circle. All other points (squares and triangle) are mapped to the left half. The yellow triangle shows how φ maps real numbers with large absolute value close to the point -1 on the unit circle.*

Given a symmetric $T \in \mathbb{R}^{n\times n}$, define

$$\varphi(T) = (iI - T)(iI + T)^{-1} = (iI + T)^{-1}(iI - T).$$

[1] The transformation used by von Neumann is $-\overline{\varphi(z)}$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

By direct computation we find that $\varphi(T)^H\varphi(T) = I$, so $\varphi(T)$ is unitary. If the eigenvalues of T are the real numbers $\lambda_1, \ldots, \lambda_n$, then the eigenvalues of $\varphi(T)$ are the numbers $\varphi(\lambda_1), \ldots, \varphi(\lambda_n)$ on the unit circle.

Let $QR = (iI - T)$ be a QR factorization of $(iI - T)$ such that the diagonal entries of R are real. Since $(iI + T) = -\overline{(iI - T)}$ we have that $(iI + T) = -\overline{QR}$, and we can now write $\varphi(T)$ using Q and R

$$\varphi(T) = -QR\overline{R}^{-1}Q^T.$$

Since Q, Q^T, and $\varphi(T)$ are unitary, we know that the product $R\overline{R}^{-1}$ must also be unitary. This, together with the fact that $R\overline{R}^{-1}$ is upper triangular, implies that $R\overline{R}^{-1}$ is a diagonal matrix. Since the diagonal entries of R are also real we have that $R\overline{R}^{-1} = I$. This gives the following factorization for $\varphi(T)$

$$\varphi(T) = -QQ^T.$$

The unitary matrix Q will always have complex entries, and thus Q^T is not the inverse of Q. We will show that Q requires only $O(n)$ storage and can be computed in $O(n)$ operations.

With the matrix $\varphi(T)$ in hand we now wish to compute its eigenvalues, which we will later use to recover the eigenvalues of T. To compute the eigenvalues of $\varphi(T)$ we use a variant of the implicitly shifted unitary QR algorithm first proposed by Gragg [8]. This algorithm reduces a unitary upper Hessenberg matrix to diagonal form in $O(n^2)$ operations using Francis's implicitly shifted QR algorithm [4, 5]. Since the product $-QQ^T$ is unitary but not upper Hessenberg, we must first reduce it to upper Hessenberg form. We will show that this reduction can be done stably in $O(n^2)$ operations. More precisely, we can compute an upper Hessenberg matrix U such that

$$U = Y^H\varphi(T)Y \tag{2.1}$$

for some unitary Y. Our algorithm does not actually compute Y.

It is well known that unitary upper Hessenberg matrices can be represented by $O(n)$ parameters. One such representation uses a factorization into $n - 1$ rotations and a diagonal matrix. Once U is in such a compressed form, the unitary QR algorithm of Gragg [8] can now be applied to U. This algorithm iteratively computes a unitary diagonal matrix Σ such that

$$Z^HUZ = \Sigma$$

for some unitary Z. Thus, the diagonal entries of Σ are the eigenvalues of $\varphi(T)$. The algorithm does not actually compute the transforming matrix Z. Letting $W = YZ$ we get the eigendecomposition

$$\varphi(T) = W\Sigma W^H.$$

To recover the eigenvalues of T one simply needs to apply φ^{-1} to the eigenvalues of $\varphi(T)$. It is straightforward to show that the inverse of φ is

$$\varphi^{-1}(z) = i(1 - z)(1 + z)^{-1}.$$

Applying this to $\varphi(T)$ we get

$$T = \varphi^{-1}(\varphi(T)) = W\varphi^{-1}(\Sigma)W^H,$$

which will be useful in our backward error analysis. The steps described above are summarized in Algorithm 1. As we will see in Section 3, Algorithm 1 is normwise backward stable

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 1 Symmetric eigenvalues via unitary QR.

Require: T real, symmetric, and tridiagonal
1) compute $QR = (iI - T)$, which gives $\varphi(T) = -QQ^T$ (not formed explicitly)
2) compute unitary upper Hessenberg U such that $U = Y^H \varphi(T) Y$
3) compute unitary diagonal Σ such that $\Sigma = Z^H U Z$
4) compute $\varphi^{-1}(\Sigma)$ to get the eigenvalues of T

whenever the norm of T is close to 1. To avoid instabilities, our implementation first computes $\alpha > 0$ such that $\alpha\|T\|_2 \approx 1$. There are many ways to calculate such an α, and we have opted to use a combination of Gershgorin's theorem and Newton's method which guarantees that we can find a good α in $O(n)$ operations. Gershgorin's theorem states that using the maximum row sums over all rows is an upper bound for $\|T\|_2$. Since every entry T_{ij} is bounded by $\|T\|_2$, the sum of the three non-zero entries in each row can be at most $3\|T\|_2$. We consider this computation an important part of a stable implementation but not necessary for the description of our algorithm.

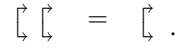
2.1. Unitary core transformations. Steps 1–3 in Algorithm 1 are implemented using unitary core transformations. A *core transformation* is any $n \times n$ matrix that is equal to the identity everywhere except for a 2×2 diagonal block, and a *unitary core transformation* is any core transformation that is also unitary. In particular, we use unitary core transformations that are also complex Givens rotations but have a real sine. The following matrix Q_2 illustrates such a core transformation for $n = 5$

$$Q_2 = \begin{bmatrix} 1 & & & & \\ & c_2 & -s_2 & & \\ & s_2 & \overline{c_2} & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}, \qquad c_2 \in \mathbb{C}, s_2 \in \mathbb{R}.$$

The subscript 2 means that the nontrivial 2×2 diagonal block is located at the intersection of the rows and columns 2 and 3 of Q_2. Since Q_2 is also a rotation, the entries c_2 and s_2 satisfy $|c_2|^2 + s_2^2 = 1$. Such a matrix can be represented by 3 real numbers, the real and imaginary parts of c_2 and the real number s_2, and by an integer locating the nontrivial diagonal block, in this case the number 2.

To simplify the presentation we denote a core transformation graphically using an arrowed bracket. For example, given the 5×5 matrix A, the product $Q_2 A$ is denoted graphically as

The arrows point to the rows of A that are affected by Q_2. Using this notation it is easy to see that two core transformations Q_i and Q_j commute if $|i - j| > 1$. When two core transformations act on the same two rows or columns, their product is also a core transformation. The multiplication of two such core transformations is known as a *fusion*



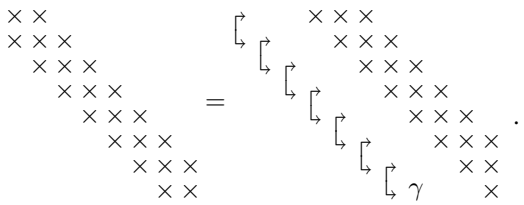
Another important operation is called a *turnover*. A turnover takes the product of three core transformations $A_i B_{i+1} C_i$ and refactors them as $\tilde{A}_{i+1} \tilde{B}_i \tilde{C}_{i+1}$. This can always be done

ETNA
Kent State University and
Johann Radon Institute (RICAM)

stably when the core transformations are unitary. Furthermore, the product of three rotations with real sine can be refactored as a product of three rotations with real sine. Graphically, a turnover looks as follows

Unitary core transformations, along with these basic operations, are the building blocks of all the algorithms in this section.

2.2. Computing $\varphi(T)$. In Step 1 of Algorithm 1 we compute the factorization $-QQ^T$ of $\varphi(T)$ by first computing the QR decomposition of $iI-T$. The matrix $(iI-T)$ is tridiagonal, so its QR decomposition can be written as a descending sequence of core transformations and a banded upper triangular R. For $n=8$, which will be our running example, we have

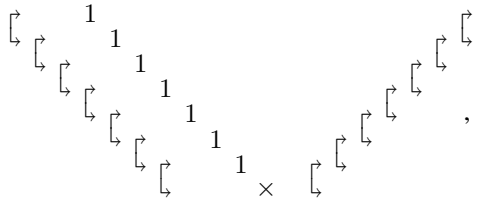


Since T is real, the matrix $(iI-T)$ has a strictly real subdiagonal, and using rotations with a real s to compute Q guarantees that all but the (n,n) diagonal entries of R are real. To make the last entry of R real we can use the core transformation $\Gamma_n = \operatorname{diag}\{1,\dots,1,\gamma\}$ with a unimodular complex number γ. We denote this core transformation by γ in the figure above.

Each subdiagonal entry of $(iI-T)$ can be zeroed using one core transformation, which can be computed and stored in $O(1)$ costs. Since there are only $n-1$ subdiagonals, the $-QQ^T$ factorization of $\varphi(T)$ can be computed in $O(n)$ time and storage. Our implementation gives the factorization

$$\varphi(T) = Q_1 Q_2 \cdots Q_{n-1} D Q_{n-1}^T \cdots Q_2^T Q_1^T,$$

where $D = \Gamma_n^2$. Using the bracket notation, the matrix $\varphi(T)$ can be depicted as (for $n=8$)

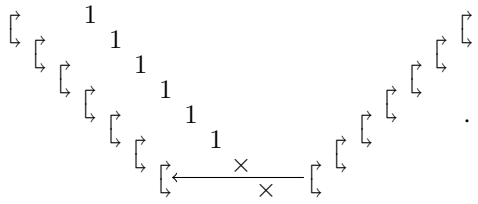


where $\times = \gamma^2$. Note that we keep $\varphi(T)$ in this factored form; we never compute it explicitly.

2.3. Reduction to upper Hessenberg form. Once the $-QQ^T$ factorization of $\varphi(T)$ is computed, it must be reduced to upper Hessenberg form before the unitary QR algorithm can be applied. This can be done by a core chasing routine similar to the one used in the unitary QR algorithm in [3]. We aim to represent the unitary Hessenberg matrix as a product of rotations with real sine and a diagonal matrix with unimodular entries.

The leftmost core transformation of Q^T can be fused into the rightmost core transformation of Q,

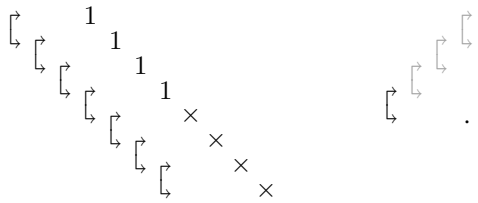
ETNA
Kent State University and
Johann Radon Institute (RICAM)



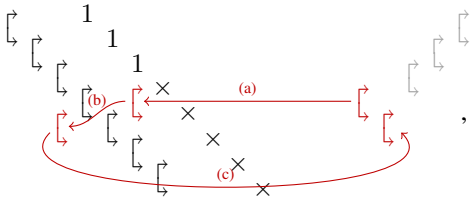
This changes the $(n-1)^{\text{st}}$ and the n^{th} entry of the diagonal matrix. However, all nonzero entries of the diagonal matrix remain unimodular.

Before the remaining core transformations can be fused, they have to be chased to the bottom of the matrix. Let us now assume that we have merged three rotations and that $i = 4$. The i^{th} core transformation has to be chased down $(n-i-1)$ rows. Chasing a core transformation down one row is the same for all rows, hence we describe only one such step.

Ignoring the other core transformations (in gray) in the rightmost sequence, we have only one "misfit" core transformation. This can be depicted as



We have the identical picture in the unitary QR misfit chasing algorithm, hence we can do the same here. We first swap the misfit with the diagonal; see arrow (a) in the following figure. This changes two diagonal entries in D. Then we perform a turnover (b) after which the misfit is in the row below. The turnover changes two rotations in Q. Now a similarity transformations (c) brings the misfit back to the right-hand side of the diagonal. The chasing can be depicted as



where we mark the misfit and its path in red. We repeat the chasing until the misfit is in the last row and can be fused with Q_{n-1}.

We repeat the process for the remaining core transformations in the rightmost sequence. Each of them has to be chased down one more row than the previous one. In total $\frac{(n-2)(n-1)}{2}$ turnovers are necessary, which need $O(n^2)$ floating point operations. This corresponds to roughly $\frac{n}{2}$ misfit chasings on a unitary Hessenberg matrix of dimension $n \times n$. The product of all the similarity transformations forms the matrix Y in (2.1).

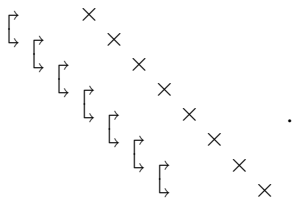
REMARK 2.1. The core transformations in the rightmost sequence can also be chased to the top of the matrix. At the top, the core transformation can be fused into the leftmost core transformation. By chasing the lower half of the core transformation to the bottom and the upper half to the top, the number of turnovers required for the reduction to Hessenberg form can be reduced by roughly a factor of two.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

2.4. Solving the unitary eigenvalue problem. After removing all of the core transformations in the rightmost sequence, the resulting unitary upper Hessenberg matrix is the product of core transformations and a diagonal matrix. The matrix U has the form

$$U = Q_1 Q_2 \cdots Q_{n-1} D,$$

where all Q_i's and all diagonal entries in D have been chased during the reduction to Hessenberg form. The matrix U can be depicted as



Our implementation of Gragg's unitary QR algorithm works efficiently on this structure and can solve the problem in $O(n^2)$ operations [3]. This implementation, as well as the one for the $\varphi(T) = -QQ^T$ factorization, are available as part of eiscor [2].

3. Assessing stability. In this section we examine the stability of Algorithm 1. The main result is that Algorithm 1 is normwise backward stable when the norm of T is close to 1. When the norm of T is far from 1, the backward error grows at most quadratically with the norm of T. To prove this stability result we first show that the errors in Steps 1-3 of Algorithm 1 are small, then we show that in Step 4 we can push this error back onto T to get the final backward stability result.

The first step of the algorithm involves computing the $-QQ^T$ factorization of $\varphi(T)$ which in turn requires the QR factorization of $(iI - T)$. In floating point arithmetic, the matrix Q will have some error, which leads to an error in the factorization of $\varphi(T)$. Lemma 3.1 gives an upper bound on the error in the computed $-QQ^T$ factorization of $\varphi(T)$.

LEMMA 3.1. *Let $T \in \mathbb{R}^{n\times n}$ be a symmetric tridiagonal matrix. Then there exists a positive constant C such that*

$$\|\varphi(T) - \hat{\varphi}(T)\|_2 \leq \begin{cases} 2Cu + O(u^2), & \|T\|_2 < 1, \\ C(1 + \|T\|_2)u + O(u^2), & \|T\|_2 \geq 1, \end{cases}$$

where $\hat{\varphi}(T) = -\hat{Q}\hat{Q}^T$ is the result of Step 1 of Algorithm 1 in floating point arithmetic and where u is the unit round-off.

Proof. In Section 2.2 we showed that computing the $-QQ^T$ factorization of $\varphi(T)$ requires only the matrix Q from the QR factorization of $(iI - T)$, so we will first give an upper bound on the computed Q. For convenience let $A = (iI - T)$. From [9, Theorem 19.10] we know that the QR factorization of A is backward stable, so there exists a positive constant C_1 and a matrix ΔA such that the computed $\hat{Q}$ satisfies

$$A + \Delta A = \hat{Q}\hat{R}, \quad \|\Delta A\|_2 \leq C_1 \|A\|_2 \, u.$$

Now let $\Delta Q = Q - \hat{Q}$. From [10, Theorem 1.1] we know there exists a positive constant C_2 such that

$$\|\Delta Q\|_2 \leq C_2 \kappa_2(A) \frac{\|\Delta A\|_2}{\|A\|_2},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$ is the condition number of A. Combining this with the backward stability result gives the bound

$$\|\Delta Q\|_2 \leq C_1 C_2 \kappa_2(A) u.$$

Now we can use the error bound for Q to bound the error of the computed $\hat{\varphi}(T)$,

$$\hat{\varphi}(T) = -\hat{Q}\hat{Q}^T = \varphi(T) - Q\Delta Q^T - \Delta Q Q^T + O(\Delta^2),$$

where $O(\Delta^2)$ summarizes higher-order terms of the perturbation. Subtracting $\varphi(T)$ from both sides and taking norms we have

$$\|\varphi(T) - \hat{\varphi}(T)\|_2 \leq 2\|\Delta Q\|_2 + O(u^2),$$

and substituting the appropriate bound gives

$$\|\varphi(T) - \hat{\varphi}(T)\|_2 \leq 2C_1 C_2 \kappa_2(A) u + O(u^2).$$

Now we use the fact that $\kappa_2(A)$ is bounded

$$\kappa_2(A) = \kappa_2(iI - T) \leq 1 + \|T\|_2,$$

(the smallest singular value of $iI - T$ is at least 1, recall that T is real and symmetric) and take $C = 2C_1C_2$ to get the final result

$$\|\varphi(T) - \hat{\varphi}(T)\|_2 \leq \begin{cases} 2Cu + O(u^2), & \|T\|_2 < 1, \\ C(1 + \|T\|_2)u + O(u^2), & \|T\|_2 \geq 1. \end{cases} \qquad \square$$

Lemma 3.1 states that computing the $-QQ^T$ factorization of $\varphi(T)$ using the QR factorization of $(iI - T)$ gives a relative error that depends linearly on the norm of T. The next lemma gives a bound on the relative backward error of our algorithm in terms of the error in Steps 1–3 and the norm of T.

LEMMA 3.2. *Let $T \in \mathbb{R}^{n\times n}$ be a symmetric tridiagonal matrix, and let*

$$\varphi(T) = (iI - T)(iI + T)^{-1}.$$

Let $\tilde{\varphi}(T)$ be a small perturbation of $\varphi(T)$, and define $\tilde{T} = \varphi^{-1}(\tilde{\varphi}(T))$. Then

$$\frac{\|T - \tilde{T}\|_2}{\|T\|_2} \leq \begin{cases} \left(1 + \|T\|_2^{-1}\right)\|\Delta\varphi(T)\|_2 + O\left(\|\Delta\varphi(T)\|_2^2\right), & \|T\|_2 < 1, \\ \left(1 + \|T\|_2\right)\|\Delta\varphi(T)\|_2 + O\left(\|\Delta\varphi(T)\|_2^2\right), & \|T\|_2 \geq 1, \end{cases}$$

where $\Delta\varphi(T) = \tilde{\varphi}(T) - \varphi(T)$.

Proof. From the definition of φ^{-1} we have

$$\tilde{T} = \varphi^{-1}(\varphi(T) + \Delta\varphi(T)) = i(I - \varphi(T) - \Delta\varphi(T))(I + \varphi(T) + \Delta\varphi(T))^{-1}.$$

Given an $n \times n$ invertible matrix A, the first-order perturbation of $(A + \Delta A)^{-1}$ for a small perturbation ΔA is

$$(A + \Delta A)^{-1} = A^{-1}(I - \Delta A A^{-1}) + O(\Delta^2).$$

Using this fact we have

$$\tilde{T} = [i(I - \varphi(T)) - i\Delta\varphi(T)](I + \varphi(T))^{-1}[I - \Delta\varphi(T)(I + \varphi(T))^{-1}] + O(\Delta^2),$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

and combining this with $T = i(I - \varphi(T))(I + \varphi(T))^{-1}$ gives

$$\tilde{T} = [T - i\Delta\varphi(T)(I + \varphi(T))^{-1}][I - \Delta\varphi(T)(I + \varphi(T))^{-1}] + O(\Delta^2).$$

Expanding the product and combining terms simplifies to

$$\tilde{T} = T - (iI + T)\Delta\varphi(T)(I + \varphi(T))^{-1} + O(\Delta^2).$$

Subtracting T from both sides, taking norms and dividing by $\|T\|_2$ gives

$$\frac{\|T - \tilde{T}\|_2}{\|T\|_2} \leq \|(I + \varphi(T))^{-1}\|_2 \frac{\|iI + T\|_2}{\|T\|_2} \|\varphi(T) - \tilde{\varphi}(T)\|_2 + O(\|\varphi(T) - \tilde{\varphi}(T)\|_2^2).$$

To complete the proof we must bound the product $\|(I + \varphi(T))^{-1}\|_2 \, \|iI + T\|_2 \, \|T\|_2^{-1}$. First we have

$$\frac{\|iI + T\|_2}{\|T\|_2} \leq \begin{cases} 1 + \|T\|_2^{-1}, & \|T\|_2 < 1, \\ 2, & \|T\|_2 \geq 1. \end{cases}$$

To bound $\|(I + \varphi(T))^{-1}\|_2$ we use the fact that the eigenvalue of $\varphi(T)$ closest to -1 corresponds to the norm of T. This gives the following identity

$$\|(I + \varphi(T))^{-1}\|_2 = |1 + \varphi(\|T\|_2)|^{-1}.$$

For $\|T\|_2 < 1$, $\varphi(\|T\|_2)$ is in the right half of the unit circle, i.e., $|1 + \varphi(\|T\|_2)|^{-1} < 1$. When $\|T\|_2 \geq 1$ we can use φ to get the following bound

$$\|(I + \varphi(T))^{-1}\|_2 = |1 + \varphi(\|T\|_2)|^{-1} = \frac{|i + \|T\|_2|}{|i + \|T\|_2 + i - \|T\|_2|} \leq \frac{1}{2}(1 + \|T\|_2).$$

Combining these with the above results and taking $\Delta\varphi(T) = \tilde{\varphi}(T) - \varphi(T)$ gives the final result

$$\frac{\|T - \tilde{T}\|_2}{\|T\|_2} \leq \begin{cases} \left(1 + \|T\|_2^{-1}\right) \|\Delta\varphi(T)\|_2 + O\left(\|\Delta\varphi(T)\|_2^2\right), & \|T\|_2 < 1, \\ \left(1 + \|T\|_2\right) \|\Delta\varphi(T)\|_2 \ + O\left(\|\Delta\varphi(T)\|_2^2\right), & \|T\|_2 \geq 1. \end{cases}$$ ∎

Lemma 3.2 states that pushing the error back onto T depends on the error in the computed $\varphi(T)$ and the norm of T. Specifically, if $\|T\|_2 \geq 1$, then the backward error scales linearly with $\|T\|_2$, and if $\|T\|_2 < 1$, then the backward error scales inverse linearly with $\|T\|_2$. Using Lemma 3.1 and Lemma 3.2 we can now prove the main result.

THEOREM 3.3. *Let $T \in \mathbb{R}^{n \times n}$ be a symmetric tridiagonal matrix. Then the eigenvalues computed by Algorithm 1 applied to T are those of a matrix $\tilde{T}$ that is near T in the following sense: there exists a linear polynomial p_1 and a quadratic polynomial p_2 such that*

$$\frac{\|T - \tilde{T}\|_2}{\|T\|_2} \leq \begin{cases} p_1\left(\|T\|_2^{-1}\right) u + O(u^2), & \|T\|_2 < 1, \\ p_2\left(\|T\|_2\right) u \ + O(u^2), & \|T\|_2 \geq 1. \end{cases}$$

Proof. From Lemma 3.1 we know that Step 1 of Algorithm 1 produces a computed $\hat{\varphi}(T)$ such that

$$\|\varphi(T) - \hat{\varphi}(T)\|_2 \leq \begin{cases} 2C_1 u \qquad + O(u^2), & \|T\|_2 < 1, \\ C_1(1 + \|T\|_2)u + O(u^2), & \|T\|_2 \geq 1. \end{cases}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

In Steps 2 and 3 we compute an eigendecomposition of $\hat{\varphi}(T)$ using the unitary core chasing algorithm available in eiscor [2], which is based on [3]. This algorithm is backward stable, so there exists a positive constant C_2 and unitary matrices $\tilde{W}$ and $\tilde{\Sigma}$, with $\tilde{\Sigma}$ diagonal, such that

$$\tilde{\varphi}(T) = \tilde{W}\tilde{\Sigma}\tilde{W}^H \qquad \text{and} \qquad \|\hat{\varphi}(T) - \tilde{\varphi}(T)\|_2 \leq C_2 u.$$

Combining the above results we have

$$\|\varphi(T) - \tilde{\varphi}(T)\|_2 \leq \begin{cases} (2C_1 + C_2)u + O(u^2), & \|T\|_2 < 1, \\ (C_1(1 + \|T\|_2) + C_2)\, u + O(u^2), & \|T\|_2 \geq 1. \end{cases}$$

Let

$$\tilde{T} = \varphi^{-1}(\tilde{\varphi}(T)) = \tilde{W}\varphi^{-1}(\tilde{\Sigma})\tilde{W}^H.$$

The eigenvalues of $\tilde{T}$ are exactly those computed in Step 4. Setting $\Delta\varphi(T) = \tilde{\varphi}(T) - \varphi(T)$ and using Lemma 3.2 we have that

$$\frac{\|T - \tilde{T}\|_2}{\|T\|_2} \leq \begin{cases} \left(1 + \|T\|_2^{-1}\right) \|\Delta\varphi(T)\|_2 + O\left(\|\Delta\varphi(T)\|_2^2\right), & \|T\|_2 < 1, \\ (1 + \|T\|_2)\, \|\Delta\varphi(T)\|_2 \ + O\left(\|\Delta\varphi(T)\|_2^2\right), & \|T\|_2 \geq 1. \end{cases}$$

Combining the above bounds and letting $p_1(z) = (2C_1 + C_2)(1 + z)$ and $p_2(z) = (C_1(1 + z) + C_2)(1 + z)$ gives the main result

$$\frac{\|T - \tilde{T}\|_2}{\|T\|_2} \leq \begin{cases} p_1\left(\|T\|_2^{-1}\right)u + O(u^2), & \|T\|_2 < 1, \\ p_2(\|T\|_2)u \ + O(u^2), & \|T\|_2 \geq 1. \end{cases} \qquad \square$$

Theorem 3.3 states that the relative backward error in Algorithm 1 depends on the norm of T. Such a dependence is undesirable but it is not unexpected. The first step of the algorithm requires a QR decomposition of $(iI - T)$, and if T is too large, then iI and T will be out of scale and important information can be rounded away. Similarly, when T is small, so are its eigenvalues. We recover these eigenvalues from the computed eigenvalues of $\varphi(T)$, thus the recovered eigenvalues are accurate relative to 1 instead of the norm of T. Both of these situations can be avoided if one first scales T to have norm close to 1, which is why we have included such a step in our implementation.

4. Numerical experiments. In this section we report numerical experiments illustrating the stability bounds from the last section. We further report timings showing that computing eigenvalues of tridiagonal matrices via the unitary QR algorithm is comparable with LAPACK's symmetric tridiagonal QR algorithm.

The numerical experiments were performed on an Intel Core i5-3570 CPU running at 3.40 GHz with 8GB of memory. GFortran 4.8.4 was used to compile the Fortran codes. For comparison we use the symmetric QR algorithm (DSTEQR), the divide and conquer method for symmetric tridiagonal matrices (DSTEVD), and the RRR method (DSTEGR) from LAPACK 3.6.0 [1], which were built with the same compiler.

In our first example we consider the symmetric tridiagonal Toeplitz matrix with 0 on the diagonal and $-\frac{1}{2}$ on the sub- and super-diagonals. This matrix is a common test case since the eigenvalues are known exactly. The eigenvalues are $\cos(j\pi/(n+1))$, $j = 1, \ldots, n$, which lie in $[-1, 1]$. This implies that $\|T\|_2 \leq 1$ and that the eigenvalues are mapped to the right half of the unit circle.

In this example we compute only the eigenvalues and measure the runtime. For smaller examples we measure the average runtime over many runs. After recording the timings, we

ETNA
Kent State University and
Johann Radon Institute (RICAM)

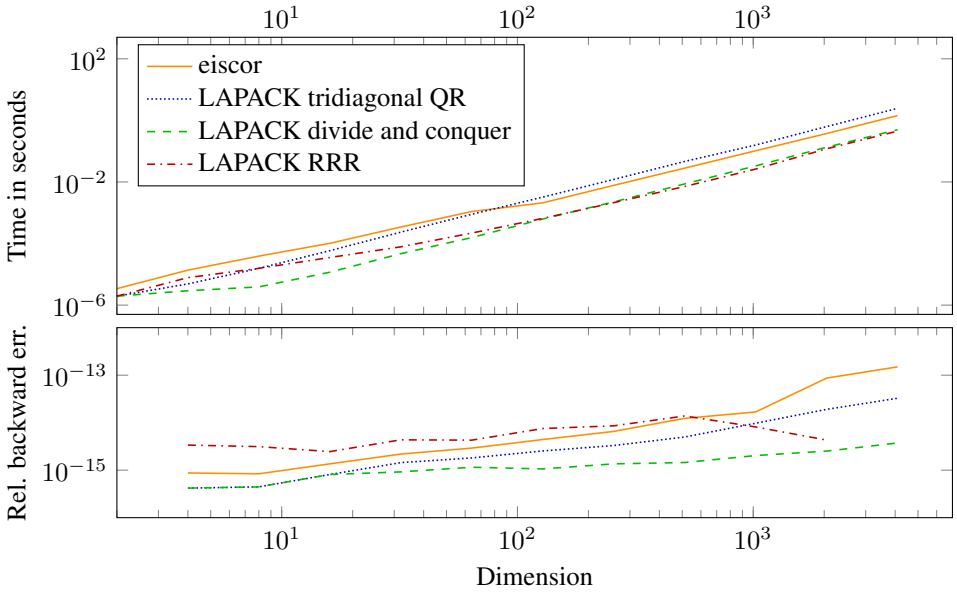


FIG. 4.1. *Runtime and relative backward error for Toeplitz matrices.*

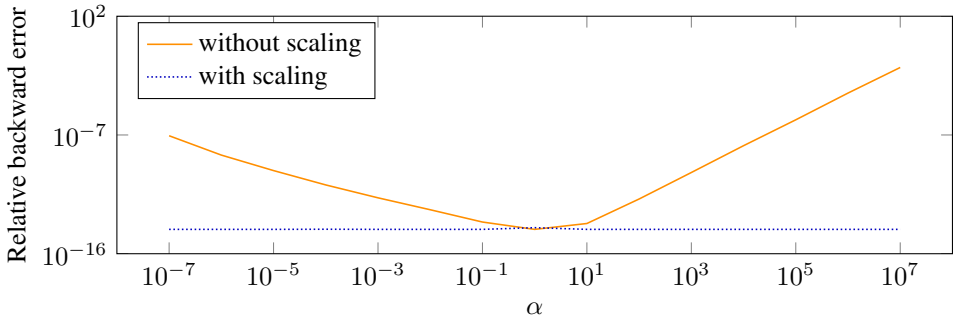


FIG. 4.2. *Backward error for $n = 512$ scaled Toeplitz matrix, with $\alpha = 10^{-7}, \ldots, 10^{7}$.*

run the experiments again computing both eigenvalues and eigenvectors. We then use the computed eigenvectors to measure the backward error. The results are shown in Figure 4.1. We observe that our algorithm is about as fast and as accurate as DSTEQR. The divide and conquer algorithm is six times faster and slightly more accurate, the RRR method is 5 times faster and about as accurate. All four algorithms need $O(n^2)$ time for computing the eigenvalues only and $O(n^3)$ if also the eigenvectors are computed, except that RRR can compute eigenvectors in $O(n^2)$.

In Section 3 we have shown that the relative backward error depends on the norm of T. Our second experiment illustrates this by scaling the $n = 512$ Toeplitz matrix from the previous example by α, for α between 10^{-7} and 10^{7}. The result in Figure 4.2 shows that when $\|T\|_2 < 1$, the backward error depends inverse linearly on $\|T\|_2$. When $\|T\|_2 \geq 1$ the dependence is quadratic. As expected, scaling the matrix based on the Gershgorin discs and the Newton correction keeps the relative backward error small.

For our third example we test our algorithm using random matrices. In Figure 4.3 we

ETNA
Kent State University and
Johann Radon Institute (RICAM)

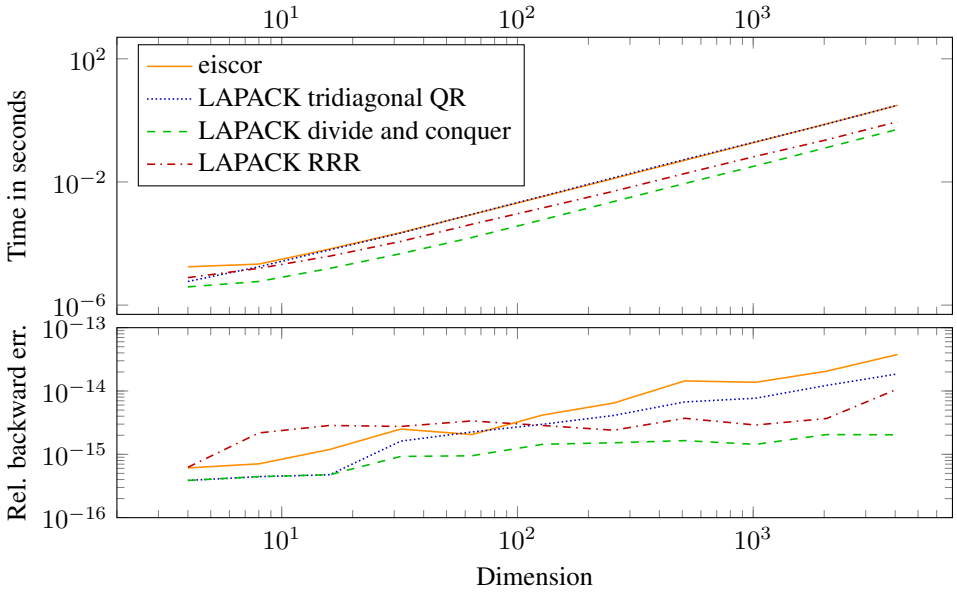


FIG. 4.3. *Runtime and backward error for random tridiagonal matrices.*

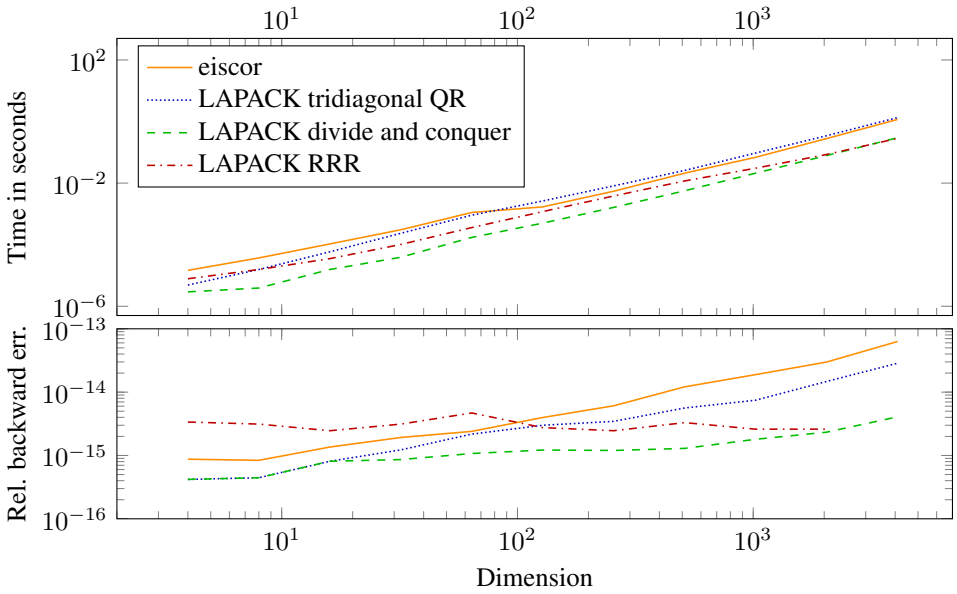


FIG. 4.4. *Runtime and relative backward error for block Toeplitz matrices.*

chose tridiagonal matrices with normally distributed entries. The results are very similar to the Toeplitz matrix used above.

In our final example we test or algorithm on matrices with clustered eigenvalues. We construct such matrices by first creating a block diagonal matrix whose diagonal blocks are 32×32 versions of the Toeplitz matrix from the first example. This matrix has 32 distinct eigenvalues that have the same multiplicity as the number of blocks. We then perturb the

ETNA
Kent State University and
Johann Radon Institute (RICAM)

off-diagonal entries at the interface of each block by 10^{-10}. The result is a set of test matrices with eigenvalues close together. The results shown in Figure 4.4 look similar to those in the previous figures.

5. Conclusions. In this paper we presented a new algorithm for solving the symmetric eigenvalue problem. This method works by converting a symmetric matrix into a unitary one and then solving the resulting unitary eigenvalue problem. We proved that under reasonable assumptions this method is backward stable and that it is comparable in terms of accuracy with current well-known implementations. It is about as fast as the symmetric, tridiagonal QR algorithm.

Acknowledgment. Jared L. Aurentz and Thomas Mach thank the Mathematisches Forschungsinstitut Oberwolfach for their generous support through the "Research in Pairs" program during which the majority of this work was carried out. The authors are also grateful to Behnam Hashemi at the University of Oxford whose careful reading of this article helped us avoiding some embarrassing mistakes. We also greatly appreciate the referees' valuable, detailed remarks.

REFERENCES

[1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, 3rd. ed., SIAM, Philadelphia, 1999.

[2] J. L. AURENTZ, T. MACH, R. VANDEBRIL, AND D. S. WATKINS, *eiscor – eigensolvers based on core transformations*, https://github.com/eiscor/eiscor, 2014–2017.

[3] ———, *Fast and stable unitary QR algorithm*, Electron. Trans. Numer. Anal., 44 (2015), pp. 327–341. http://etna.ricam.oeaw.ac.at/vol.44.2015/pp327-341.dir/pp327-341.pdf

[4] J. G. F. FRANCIS, *The QR transformation a unitary analogue to the LR transformation. I.*, Comput. J., 4 (1961), pp. 265–271.

[5] ———, *The QR Transformation. II.*, Comput. J., 4 (1962), pp. 332–345.

[6] J. N. FRANKLIN, *Matrix Theory*, Prentice-Hall, Englewood Cliffs, 1968.

[7] L. GEMIGNANI, *A unitary Hessenberg QR-based algorithm via semiseparable matrices*, J. Comput. Appl. Math., 184 (2005), pp. 505–517.

[8] W. B. GRAGG, *The QR algorithm for unitary Hessenberg matrices*, J. Comput. Appl. Math., 16 (1986), pp. 1–8.

[9] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[10] G. W. STEWART, *Perturbation bounds for the QR factorization of a matrix*, SIAM J. Numer. Anal., 14 (1977), pp. 509–518.

[11] J. V. NEUMANN, *Allgemeine Eigenwerttheorie Hermitescher Funktionaloperatoren*, Math. Ann., 102 (1930), pp. 49–131.